LETTER Communicated by Richard Lippmann

A Fast Histogram-Based Postprocessor That Improves Posterior Probability Estimates

Wei Wei*
Todd K. Leen
Etienne Barnard†
Department of Computer Science and Engineering, Oregon Graduate Institute of Science and Technology, Portland, OR 97291-1000, U.S.A.

Although the outputs of neural network classifiers are often considered to be estimates of posterior class probabilities, the literature that assesses the calibration accuracy of these estimates illustrates that practical networks often fall far short of being ideal estimators. The theorems used to justify treating network outputs as good posterior estimates are based on several assumptions: that the network is sufficiently complex to model the posterior distribution accurately, that there are sufficient training data to specify the network, and that the optimization routine is capable of finding the global minimum of the cost function. Any or all of these assumptions may be violated in practice.

This article does three things. First, we apply a simple, previously used histogram technique to assess graphically the accuracy of posterior estimates with respect to individual classes. Second, we introduce a simple and fast remapping procedure that transforms network outputs to provide better estimates of posteriors. Third, we use the remapping in a real-world telephone speech recognition system. The remapping results in a 10% reduction of both word-level error rates (from 4.53% to 4.06%) and sentence-level error rates (from 16.38% to 14.69%) on one corpus, and a 29% reduction at sentence-level error (from 6.3% to 4.5%) on another. The remapping required negligible additional overhead (in terms of both parameters and calculations). McNemar's test shows that these levels of improvement are statistically significant.

1 Introduction

Given an input x, the output values $y_i(x)$ of a trained neural network classifier are often regarded as estimates of posterior class probabilities $P(C_i|x)$. This association is implied by theorems showing that global (over the space

* Currently at Mentor Graphics Corporation, Wilsonville, OR.
† Currently at Speechworks International, Boston, MA.

of networks) minima of various cost functions correspond to network outputs that are the desired posteriors (Duda & Hart, 1973; Hampshire & Perlmutter, 1990; Bourlard & Morgan, 1994). A functionally rich network trained on a sufficiently large database, using an optimizer likely to find good optima, presumably can approach the results of these theorems.

In practice these assumptions are not satisfied, and trained networks need not provide good estimates of posterior class probabilities. To address the problem, we develop a fast postprocessing procedure that remaps network outputs through a simple function to obtain better estimates of posterior class probabilities. We then show that the remapping improves the performance of a complete speech recognition system that follows the network output with a standard Viterbi search (a hybrid neural network/hidden Markov model system). We show that the improvement gained *cannot* be matched by adjustment of the transition probabilities in the Markov model. Thus, improving the posterior estimates is critical to improved performance. We demonstrate significant performance improvement on two different speech corpora with differing statistics.

Postprocessing network outputs is not a new idea. Denker and Le Cun (1991) proposed a remapping technique with the same aim considered here. Their method is based on the notion that one ought to estimate the conditional joint distribution in the output space $p(y_1, \ldots, y_k \mid C_j)$ and use these to estimate the class posteriors $p(C_j \mid x)$ (with an intermediate distribution on the outputs $p(y_i \mid x$, training data) that follows from the usual Bayesian formulation). Application of their procedure could be cumbersome in the high-dimensional output spaces typical in speech recognition systems. Our networks, for example, have several hundred outputs corresponding to context-dependent phone classes. The method we develop deals instead with only the marginal densities in the output space (one unit at a time) and is therefore applicable even for very large output dimension.[1]

The method we propose uses a set of univariate histogram-like plots (one for each output class) that graphically portray the discrepancy between the network outputs and true posteriors (Wei, Barnard, & Fanty, 1996). Although independently developed for this study, these plots are similar to the reliability diagrams used by Dawid (1986) to discuss accuracy of probabilistic forecasting (e.g., precipitation probability in weather forecasts). Such plots were also used by Lippmann and Shahian (1997) to assess calibration accuracy of various neural networks employed as posterior estimators for a medical risk prediction task. Bourlard and Morgan (1994) and Ripley (1996) used similar plots as calibration to illustrate or determine whether posterior probability estimates are indeed good estimates.[2]

[1] We are grateful to one of the reviewers for pointing out Denker and Le Cun's previous work on this issue.

[2] We are grateful to one of the reviewers for pointing out previous work on this issue by Dawid; Bourlard and Morgan; Ripley; and Lippmann and Shahian.

We extend the plots from an assessment tool to a recalibration tool, proposing a simple class of univariate remapping functions that bring the network outputs into closer agreement with the true posteriors. Finally, we apply the approach to real speech recognition systems that integrate the remapped neural acoustic front end with a Viterbi search.

In section 2 we develop the histogram display and introduce the remapping in section 3. Section 4 describes our experimental results on two speech recognition corpora, and we close with the summary in section 5.

2 Empirical Measurement of Neural Network Output Estimates

We use a histogram-like technique to measure the accuracy of network estimates of posterior probabilities. Suppose that the ith output assumes the value $y_i(x)$, given an input pattern x. Since this output is supposed to estimate the posterior class probability, we should have

$$y_i(x) \approx P(C_i \mid x).$$

Suppose that we could obtain a very large set of patterns $X = \{x^{(1)}, x^{(2)}, \ldots\}$ that all give rise to the same output value y_{i_0} on the ith node. Then the frequency of occurrence of class C_i (the frequency with which the ith node has target 1) for these samples, denoted $\Gamma_i(X)$, ought to be roughly equal to the posterior

$$\Gamma_i(y_{i_0}) \equiv \frac{\text{Number of samples with output } y_{i_0} \text{ that belong to } C_i}{\text{Total number of samples with output } y_{i_0}}$$
$$\approx P(C_i | x \in X). \tag{2.1}$$

We call this quantity the *matching frequency*.

Of course, we are not able to obtain multiple examples with exactly the same output value y_{i_0}. Instead we collect observed values in a range $[y_{i_0} - \delta_1, y_{i_0} + \delta_2]$ and plot on the vertical scale the fraction $\Gamma_i(y_{i_0})$ of these observed values for which the target on node i is 1. This vertical distance is an empirical approximation of the posterior class probability $p(C_i|y_i(x)) \equiv p(C_i|x)$. Thus, we construct a histogram of the frequency with which node i's target is 1 for each of a number of bins of output value range. This is simply a histogram of the frequency with which the sample targets match the class label C_i.

For the ideal posterior estimator, the histogram points will lie close to the diagonal line. Anywhere the histogram falls below the diagonal line, the matching frequency (and hence the empirical estimate of the posterior) is less than the network output; that is, the network has overestimated the posterior probability. Similarly, when the histogram lies above the diagonal, the network has underestimated the posterior.

The two histograms in Figure 1 were made from experiments on spoken digit recognition on the OGI (Oregon Graduate Institute of Science and Technology) Numbers Corpus. (The data and corresponding recognition task are discussed in detail in section 4.) The plots confirm that networks

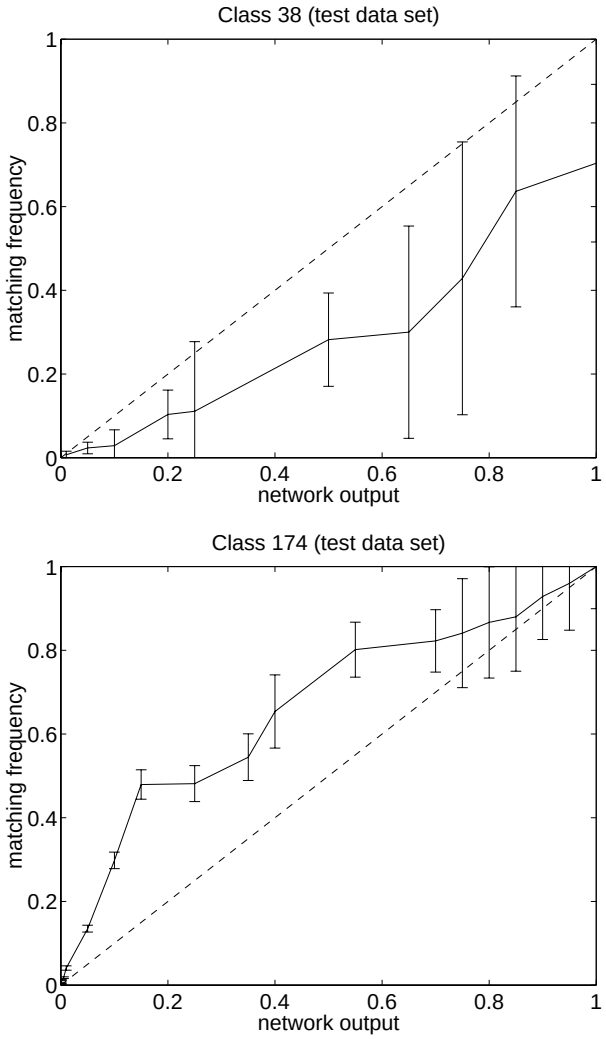

Figure 1: Example histograms.

are not ideal posterior estimators. For example, when the network output equals 0.4, the matching frequencies are 0.21 (posterior overestimated by the network output) and 0.65 (posterior underestimated by the network output), respectively. We use χ^2 tests to evaluate the calibration accuracy, as in Lippmann and Shahian (1997). The resulting χ^2 tests are significant at the 0.05 level (indicating poor calibration accuracy) for both classes (with 19 and 37 degrees of freedom, respectively). Figure 1 also shows error bars ($\pm$ one

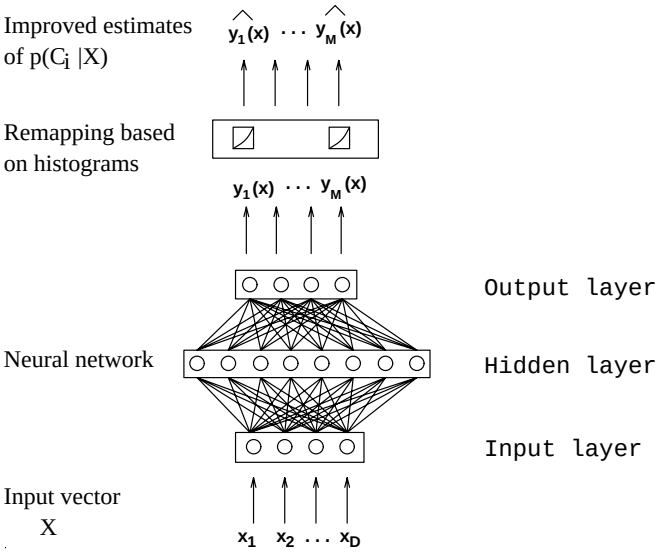

Figure 2: Neural network with univariate output remapping.

σ for the binomial distribution of matching counts over the total sample counts in each bin). For this example, the network is a three-layer perceptron network consisting of 56 inputs, 200 hidden nodes, and 209 outputs, trained with stochastic backpropagation (Werbos, 1974; Parker, 1985; Rumelhart, Hinton, & Williams, 1986), using the cross-entropy cost function (Hopfield, 1987; Baum & Wilczek, 1988; Solla, Levin, & Fleisher, 1988; Hinton, 1989).

3 Remapping Neural Network Outputs Based on Histograms

Our postprocessing strategy is to remap the outputs of neural networks to improve the estimation accuracy of posterior probabilities. The histograms described in the previous section provide an indication of the discrepancy between the network outputs and the true posteriors. With the remapping, the system appears as in Figure 2. As before, $x \in R^D$ is the input and $y_i(x),\ i = 1, \ldots, M$ are the outputs, corresponding to the classes $\{C_i\colon\ i = 1, \ldots, M\}$. The remapped outputs are denoted $\hat{y}_i$. We note that the remapping functions are univariate, that is, $\hat{y}_i = f(y_i)$.

Because of limited data, if we use a fixed-space bin set for all classes, we may find some unpopulated bins. This type of histogram is not smooth enough for remapping. In particular, we want the histograms to be monotonically increasing with increasing y_i. To accomplish this, we dynamically adjust the bins as follows. We start with a fixed-space bin set. We divide empty bins into two halves and merge each half with its left or right neigh-

boring bin. We merge a nonempty bin with its right neighbor if this neighbor contains a lower value of matching frequency. We repeat the merging process until we obtain a monotonically increasing histogram. This bin adjustment is carried out separately for each output unit.

Next we construct the remapping function. The new outputs $\hat{y}_i$ are supposed to have matching frequency histograms that are diagonal; thus ideally we would like to have

$$\Gamma_i(y_i) = \hat{y}_i(y_i) \equiv f(y_i), \tag{3.1}$$

where Γ_i is the matching frequency defined in equation 2.1 and the paragraph following. Hence the desired remapping function is simply the smooth function that best approximates the original histogram points.

We have tried several functions for the remapping and achieved our best results with a function that is log-linear for output values below s (which itself is determined during the fitting) and linear for output values above s:

$$f(y_i) = \begin{cases} a * y_i^b, & \text{if } y_i \le s; \\ c * (y_i - s) + a * s^b, & \text{otherwise.} \end{cases}$$

The parameters a, b, c, s are fit as follows. We quantize the possible values of s, $0 \le s \le 1$, in equal intervals of 0.05. For each candidate value of s, the parameters a, b, and c are chosen so that $f(y_i;\ a, b, c)$ is the least-squares fit to the matching frequency $\Gamma_i(y_i)$. Among these possible candidate functions (differing by the cross-over point s), we choose the one that minimizes the mean absolute difference between $\hat{y}_i$ and Γ_i. This determines all the function parameters.

The combination of the log-linear function in the first interval $[0, s]$ and the linear function in the second interval $[s, 1]$ offers a more flexible map than the linear fitting in the whole range $[0, 1]$. We use histograms calculated on the cross-validation data, not the training data, to specify the remapping parameters.

Because the transformation function is very simple, applying remapping during recognition has a negligible effect on the total computation time. The remapping transformation requires only four additional parameters for each output node, a negligible number in comparison to the number of network weights.

Since the functional form is rather smooth, remapping cannot effectively produce a new histogram close to the diagonal if the initial (monotonic) histogram has very few bins. Hence, we set a threshold k and do the remapping on those output nodes (classes) whose histograms contain more than k bins; otherwise, we do not. On the cross-validation data, we found that a threshold $k = 15$ reliably picked those histograms for which the remapping is effective.

Invariably some of the classes will be trained better than others. Classes for which the initial posterior estimates are very poor cannot be improved by

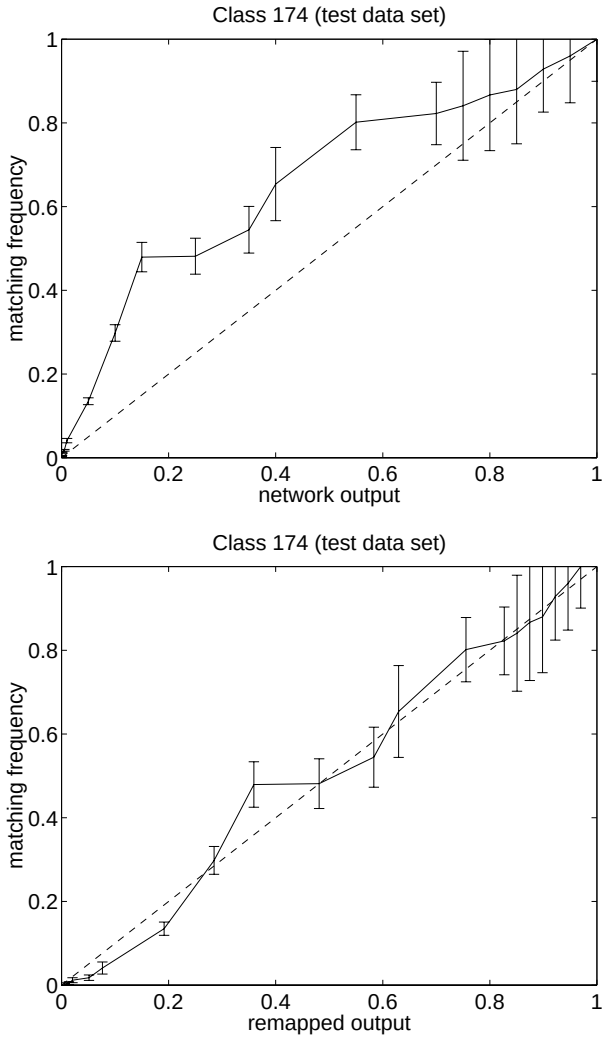

Figure 3: Histograms before and after remapping.

our simple remapping. To weed out poorly trained classes, we require that the remapped output be greater than 0.9 when the raw network output is unity ($f(1) > 0.9$). This avoids remapping when there is initially a large error in the high-probability end of the range. We also require that the remapping decreases the error rate (at the frame level) on the cross-validation data.

Figure 3 shows the histograms of matching frequency on the test data (shown by the cross points) using the raw neural network estimates and

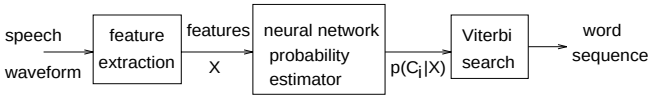

Figure 4: Neural network-based speech recognition system.

remapped output estimates (with remapping parameters $s = 0.55, a = 1.60, b = 0.57$, and $c = 0.48$ for this example), respectively. The histogram based on the remapped outputs is closer to the diagonal than the histogram generated by the raw network outputs. Thus, the remapped output gives a more accurate estimate of posterior probabilities than the raw outputs. The resulting χ^2 test is not significant (with 37 degrees of freedom) at the 0.05 level, which also indicates a good calibration accuracy.

4 Experiments on Speech Recognition Tasks

We want to improve the recognizer's posterior estimates so that the performance of a complete system is improved. Figure 4 shows a complete speech recognition system. The class posterior estimates provided by the neural network (or remapped network in our case) are used by a Viterbi search (Morgan & Bourlard, 1995) to find the most probable alignment path.

We used two telephone speech corpora for our experiments: (1) recognition of digit utterances from the OGI Numbers Corpus and (2) recognition of year utterances from the OGI Census Year Corpus. The speech data for the two tasks are collected from different speakers and different handsets.

4.1 Recognition of Digits from the OGI Numbers Corpus. The OGI Numbers Corpus (Cole, Noel, Lander, & Durham, 1995) is a telephone speech corpus that contains fluent numbers, such as house numbers from an address. The utterances in this corpus were taken from other telephone speech data collections completed at OGI. In most data collections, the callers were asked to leave their telephone number, birthdate, or zip code at some point. Also, the callers would occasionally leave numbers in the midst of another utterance. The numbers in these were extracted and included in the numbers corpus. Examples of digit utterance are "oh one oh nine three oh," "three eight four," and "two eight zero oh seven." The digit utterances are composed of sequences with lengths between 1 and 12 digits. Thus, the task contains both short and long digit sequences.

For this speech recognition task, the neural network consists of 56 input nodes, 200 hidden nodes, and 209 output nodes that correspond to 209 context-dependent phonetic units (described below). In the front end of this speech recognition system, the incoming speech waveform is converted to frame-based perceptual linear prediction cepstrum coefficients (Hermansky, 1990). The extracted features are then used as the input vector to the

neural network estimator. The network outputs estimate class probabilities that are used by a Viterbi search to find the most probable alignment path. Word models are used to define the possible state transitions from one context-dependent unit to the next context-dependent unit within words. To get the utterance (sentence), a grammar constrains the set of possible next words.

The output representation requires some explanation. A phoneme is an abstract linguistic unit that forms the basis for writing down a language (changing a phoneme changes a word). The acoustic realization of a phoneme (called a phone) can have a wide range of variation. Some of this variation is the result of varying left and right context. In our experiments, we use context-dependent phonetic modeling by using phonetic units that include left or right contexts. For example, the word *nine* surrounded by silence can be modeled as *sil!<n* (silence followed by nasal *n*), *n>!aI* (*n* followed by diphthong *aI*), ⟨*aI*⟩ (*aI*), *aI*! < *n* (*aI* followed by *n*) and *n* >!*sil* (*n* followed by silence), corresponding to five segment classes.[3] The word *nine* may also follow, or be followed by, one of the 10 words (digits) without interword silence. Thus, 20 more context-dependent classes can occur. Context-independent phonetic models (monophones) are poor discriminators. During speech, the vocal tract is constantly in motion. Consequently phonetic units are neither discrete, nor uniform, nor independent. Context-dependent phonetic models provide more accurate description of speech.

The utterance-initial and utterance-final silence as well as interword silence are modeled by the class ⟨*.pau*⟩. Note that ⟨*.pau*⟩ may or may not occur between two words; both options are available in the grammar. Class ⟨*.pau*⟩ plays a very important role in the recognition at both word level and sentence level. The number of word insertions and word deletions to compose a sentence is significantly influenced by the recognition of class ⟨*.pau*⟩, especially when the utterances contain different length of words and some of them are long word sequences. For example, insertion and deletion errors for the class ⟨*.pau*⟩ may cause *zero oh* and *zero* to be confused.

In our experiments, we separated the data into training, cross-validation, and test data in the ratio of 3:1:1. We use a multilayered perceptron (MLP)–based remapping model (shown as Figure 2) as the class probability estimator instead of a standard MLP. As we mentioned in section 2, the network is trained with stochastic backpropagation algorithms using the cross-entropy cost function.

There are 209 classes corresponding to the context-dependent phonetic units. The constraints discussed in the previous section drastically reduce the number of classes that are candidates for remapping. Following the procedure for producing monotonic histograms, only four classes had his-

[3] The symbols !< and >! represent the left or right context information—for example, *n* >!*aI* means "the last part of *n* before *aI*."

Table 1: Recognition Results on Digits from OGI Numbers Corpus (Evaluation on 1899–Sentence Test Set).

Estimator	Neural Network	Remapped Network	Error Reduction
Error(sentence)	16.38%	14.69%	10.32%
Error(word)	4.53%	4.06%	10.38%

tograms containing more than $k = 15$ bins and remapped outputs greater than 0.9 for raw outputs of 1.0. Doing the remapping on these classes and using the cross-validation data to select the classes whose correct classification rates are increased after remapping, one class $\langle .pau \rangle$ remains the only candidate for remapping.[4] Therefore, only four parameters (1 class $\times$ 4 parameters per class) are added for remapping. Compared with the number of the network weights, which is 53,002 ($56 \times 200 + 200 \times 209 + 2$), this is a negligible increase in the number of parameters (and also increases the computational cost of acoustic scoring negligibly).

As shown in Table 1, the remapping resulted in a 10.32% reduction in recognition errors at the sentence level and a 10.38% at the word level. A McNemar significance test (Gillick & Cox, 1989) shows that the observed differences would arise by chance on much less than 0.5% of occasions. The performance improvement is statistically significant.[5]

We also measured the correct classification rates at the frame (context-dependent phonetic unit) level on the test data. Remapping increased the correct classification rate for $\langle .pau \rangle$ 12.61%, from 51.00% to 63.61%. Because the utterances of this speech recognition task have different lengths, class $\langle .pau \rangle$ plays a very important role in word insertions and deletions. The word accuracy is computed as $100\% - \%substitutions - \%deletions - \%insertions$. In our experiments, the number of insertion plus deletion errors is decreased from 263 to 197 (the total number of words is 10,196).

To measure the histogram improvement, we compare the average mean of the absolute difference between the matching frequency and the probability estimate (either from the network output or from the remapped output) for the remapped class on the test data. The difference for remapped

[4] The remapped class $\langle .pau \rangle$ contains the largest amount of training data (20,000 training samples, whereas the average number of training samples per class is about 1513). It contains 5852 test samples, and the average number of test samples per class is about 555. Class $n >!sil$ contains the largest number of test samples—6682.

[5] The error rates reported here are typical for this database using various methods. Yan, Fanty, and Cole (1997) report 4.9% word and 16.7% sentence level error rates. Hosom and Cole (1997) report a 4.6% word error rate. Burnett and Fanty (1996) report a 6.0% word error rate. Burnett and Fanty also trained their system on the TIDIGITS database and report a significantly lower word error rate of 0.7% on that database. The OGI Number Corpus used here presents a difficult recognition task.

class ⟨*.pau*⟩ is decreased from 0.1028 (network output estimates) to 0.0252 (remapped output estimates). This result shows that the remapped outputs provide more accurate estimates of posterior probabilities than the network outputs.

One might argue that remapping this single class could be replaced by merely scaling the probability of ⟨*.pau*⟩. This is not the case. To explore this, we multiplied the probability of ⟨*.pau*⟩ by a constant factor chosen to optimize performance on the validation data. We selected 11 factors on a logarithmic scale in the range 0.8 to 4.0 (if the factor is 1.0, the probability is the network output).[6] The optimal value of the scaling factor is 1.1038. Increasing the probability of ⟨*.pau*⟩ by this factor resulted in a 16.06% error rate at sentence level and a 4.42% error rate at word level, whereas the number of insertions and deletions is 253. Finally, we tried rescaling the probability of ⟨*.pau*⟩ to obtain a histogram close to the diagonal. This results in a scaling factor 1.15. This gives the same error rate on the cross-validation data but slightly better results on the test data: 15.96% error rate at sentence level and a 4.39% error rate at word level. Either attempt shows that simply rescaling the probability for ⟨*.pau*⟩ does not achieve performance gains comparable to our remapping.

4.2 Recognition of the OGI Census Year Corpus. Our second set of experiments also used telephone speech, but examples were from the OGI Census Year Corpus, for which the training data are *not* strongly dominated by the class ⟨*.pau*⟩.[7] This corpus was created as part of a study to determine the feasibility of using an automated spoken questionnaire to collect information for the Year 2000 U.S. Census (Barnard, Cole, Fanty, & Vermeulen, 1995). The database used contains utterances of calendar years (e.g., "nineteen fifty-two" and "twenty-nine"). The length of the year utterance is either two words or three words.

We separate this data set into three parts by the ratios 3:1:1, used as training, cross-validation, and test data, respectively. There are 123 classes for the context-independent phonetic units. The neural network is a three-layer perceptron consisting of 56 input nodes, 45 hidden nodes, and 123 output nodes that correspond to 123 classes. It is trained by stochastic backpropagation, using the cross-entropy cost function. For remapping, we initially retain 13 classes that have histograms with more than $k = 15$ bins and whose remapped output is greater than 0.9 when the network output is 1.0. All 13 of these remapped classes showed improved frame-level classi-

[6] The factors are 0.8000, 0.9397, 1.1038, 1.2965, 1.5229, 1.7889, 2.1012, 2.4681, 2.8991, 3.4054, and 4.0000.

[7] There are 5000 training examples of class ⟨*.pau*⟩, while the average number of training samples of all the classes is about 1943.

Table 2: Recognition results on OGI Census Year Corpus (Evaluation on 734-Sentence Test Set).

Estimator	Neural Network	Remapped Network	Error Reduction
Error(sentence)	6.3%	4.5%	28.57%

fication rates. Hence the remapping was retained for all 13 output nodes. Therefore, 52 parameters (13 class $\times$ 4 parameters per class) are added for remapping. Compared with the number of the network weights, which is 8057 ($56 \times 45 + 45 \times 123 + 2$), the number of additional parameters required is negligible. The additional calculation used for remapping is also negligible.

As shown in Table 2, the remapping resulted in a 28.57% reduction in recognition errors at the sentence level. The results of McNemar's test show that the observed differences would arise by chance on much less than 0.5% of occasions. The performance improvement is statistically significant.

We also measured the change in frame-level classification rates as a result of the remapping. The average correct classification rate of the 13 remapped classes increased from 52.94% to 61.93% after remapping. The largest increase in correct rate was 18.99%. The correct classification rate of $\langle .pau \rangle$ is increased 6.42%, from 9.1% to 15.53% after remapping.

To measure the histogram improvement, we compare the mean of the absolute difference between the matching frequency and the probability estimate before and after remapping. The average value of the mean absolute difference of the 13 remapped classes decreased from 0.1128 (network output estimates) to 0.0555 (remapped output estimates), and the maximum decrease of such difference among the 13 classes is 0.1496, from 0.1672 to 0.0176. These results, along with the increases in frame-level classification, show that the remapped outputs provide more accurate estimates of posterior probabilities than the raw network outputs.

5 Conclusions

Histogram plots provide a quick visual assessment of the accuracy of class posterior probability estimates provided by neural networks. Simple univariate remapping functions can improve these estimates with a minimum of computation and very few additional model parameters. Our experiments on two different telephone speech recognition systems show that the remapping procedure improves performance at the frame, word, and sentence levels. We expect that this simple technique will be valuable for a broad range of applications where it is important to provide not only classification, but reliable posterior probability estimates.

Acknowledgments

We thank the referees for their helpful critique. W. W. and E. B. were supported by grant IRI-9529006 from the National Science Foundation and the Defense Advanced Research Projects Agency. T. L. was partially supported by grant ECS-9704094 from the National Science Foundation.

References

Barnard, E., Cole, R., Fanty, M., & Vermeulen, P. (1995). Real-world speech recognition with neural networks. *Proceedings of the International Symposium on Aerospace/Defense Sensing and Control and Dual-Use Photonics* (Orlando, FL).

Baum, E. B., Wilczek, F. (1988). Supervised learning of probability distributions by neural networks. In D. Z. Anderson (Ed.), *Neural information processing systems* (pp. 52–61). New York: American Institute of Physics.

Bourlard, H. A., & Morgan, N. (1994). *Connectionist speech recognition—a hybrid approach*. Norwell, MA: Kluwer.

Burnett, D., & Fanty, M. (1996). Rapid unsupervised adaptation to children's speech on a connected-digit task. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. New York: IEEE.

Cole, R. A., Noel, M., Lander, T., & Durham, T. (1995). New telephone speech corpora at CSLU. *Proceedings of the Fourth European Conference on Speech Communication and Technology, 1*, 821–824.

Dawid, A. P. (1986). Probability forecasting. In S. Kotz (Ed.), *Encyclopedia of statistical sciences* (pp. 210–218). New York: Wiley.

Denker, J. S., & Le Cun, Y. (1991). Transforming neural-net output levels to probability distributions. In R. Lippman, J. Moody, and D. Touretzky (Eds.), *Advances in neural information processing systems, 3* (pp. 853–859). San Mateo, CA: Morgan Kaufmann.

Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. New York: Wiley-Interscience.

Gillick, L., & Cox, S. J. (1989). Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 532–535).

Hampshire II, J. B., & Perlmutter, B. A. (1990). Equivalence proofs for multilayer perceptron classifiers and the Bayesian discriminant function. In D. Touretzky, J. Elman, T. Sejnowski, & G. Hinton (Eds.), *Proceedings of the 1990 Connectionist Models Summer School*. San Mateo, CA: Morgan Kaufmann.

Hermansky, H. (1990). Perceptual linear predictive PLP analysis for speech. *Journal of the Acoustic Society of America, 87*(4), 1738–1752.

Hinton, G. E. (1989). Connectionist learning procedures. *Artificial Intelligence, 40*, 185–234.

Hopfield, J. J. (1987). Learning algorithms and probability distributions in feed-forward and feed-back networks. *Proceedings of the National Academy of Sciences, 84*, 8429–8433.

Hosom, J. P., & Cole, R. A. (1997). A diphone-based digit recognition system using neural networks. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. New York: IEEE.

Lippmann, R. P., & Shahian, D. M. (1997). Coronary artery bypass risk prediction using neural networks. In *Annals Thoracic Surgery 1997* (pp. 1635–1643). Amsterdam: Elsevier.

Morgan, N., & Bourlard, H. (1995, May). Continuous speech recognition. *IEEE Signal Processing Magazine*, pp. 25–42.

Parker, D. B. (1985). *Learning logic* (Tech. Rep. No. TR-47). Cambridge, MA: MIT Center for Research in Computational Economics and Management Science.

Ripley, B. (1996). *Pattern recognition and neural networks*. Cambridge: Cambridge University Press.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1988). Learning internal representations by error propagation. In D. E. Rumelhart, J. L. McClelland, & the PDP Research Group (Eds.), *Parallel distributed processing: Explorations in the microstructure of cognition* (Vol. 1, pp. 318–362). Cambridge, MA: MIT Press.

Solla, S. A., Levin, E., & Fleisher, M. (1988). Accelerated learning in layered neural networks. *Complex Systems, 2*, 625–640.

Wei, W., Barnard, E., & Fanty, M. (1996). Improved probability estimation with neural network models. *Proceedings of the International Conference on Spoken Language Systems* (pp. 498–501).

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences*. Unpublished doctoral dissertation, Harvard University.

Yan, Y., Fanty, M., & Cole, R. (1997). Speech recognition using neural networks with forward-backward probability generated targets. In *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*. New York: IEEE.

Received August 7, 1997; accepted September 21, 1998.